Лабораторна Робота №3

Тема: Дослідження методів регресії

Посилання на гітхаб: https://github.com/ViMIMercurysMight/python-ai.git

Завдання 2.1. Створення регресора однієї змінної

*Лістинг програми*

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
# Вхідний файл, який містить дані  
input\_file = 'data\_singlevar\_regr.txt'  
  
# Завантаження даних  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Розбивка даних на навчальний та тестовий набори  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
# Тренувальні дані  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
# Тестові дані  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
# Створення об'єкта лінійного регресора  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.predict(X\_test)  
  
# Побудова графіка  
plt.scatter(X\_test, y\_test, color='green')  
plt.plot(X\_test, y\_test\_pred, color='black', linewidth=4)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
print("Продуктивність лінійної регресії:")  
print("Середня абсолютна похибка =",  
 round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Середня квадратична помилка =",  
 round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Середня абсолютна помилка =",  
 round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Пояснена оцінка дисперсії =",  
 round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 оцінка =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))

output\_model\_file = 'model.pkl'

# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nНова середня абсолютна помилка =",  
 round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

Text

Description automatically generated



Рисунок 1. Результат виконання

Ми можемо використовувавти цей спосіб для статистичного аналізу, що показує зв’язок між двома змінними. Лінійна регресія може

створити модель прогнозування за «випадковими» данами, показуючи тенденцію в них. Наприклад для цін, тривалості життя тощо.

Завдання 2.2. Передбачення за допомогою регресії однієї змінної

Варіант 1 файл: data\_regr\_1.txt

*Лістинг програми*

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
print("TEST")  
# Вхідний файл, який містить дані  
input\_file = 'data\_regr\_1.txt'  
  
# Завантаження даних  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Розбивка даних на навчальний та тестовий набори  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
# Тренувальні дані  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
# Тестові дані  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
# Створення об'єкта лінійного регресора  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.predict(X\_test)  
  
# Побудова графіка  
plt.scatter(X\_test, y\_test, color='green')  
plt.plot(X\_test, y\_test\_pred, color='black', linewidth=4)  
plt.xticks(())  
plt.yticks(())  
#plt.show()  
  
print("Продуктивність лінійної регресії:")  
print("Середня абсолютна похибка =",  
 round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Середня квадратична помилка =",  
 round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Середня абсолютна помилка =",  
 round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Пояснена оцінка дисперсії =",  
 round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 оцінка =", round(sm.r2\_score(y\_test, y\_test\_pred), 3))  
  
output\_model\_file = 'model.pkl'  
  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nНова середня абсолютна помилка =",  
 round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

Text

Description automatically generated



Рисунок 2. Резльтат виконання

Графік вийшов ідентичний через повну подібність вмісту файлу data\_singlevar\_regr.txt та файлу data\_regr\_1.txt. З графіка видно, що залишки розподілені більш-менш рівномірно відносно осі. Виходячи з R2 оцінки можна зробити висновок, що продуктивність цієї моделі машинного навчання на основі регресії є середньою

A picture containing chart

Description automatically generated

Рисунок 3. Демонстрація подібності файлів

Завдання 3. Створення багатовимірного регресора

*Лістинг програми*

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import PolynomialFeatures  
  
# Вхідний файл, який містить дані  
input\_file = 'data\_multivar\_regr.txt'  
  
# Завантаження даних  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Розбивка даних на навчальний та тестовий набори  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
# Тренувальні дані  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
# Тестові дані  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
# Створення об'єкта лінійного регресора  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.predict(X\_test)  
  
# # Побудова графіка  
# plt.scatter(X\_test, y\_test, color='green')  
# plt.plot(X\_test, y\_test\_pred, color='black', linewidth=4)  
# plt.xticks(())  
# plt.yticks(())  
# plt.show()  
  
print("Продуктивність лінійної регресії:")  
print("Середня абсолютна похибка =",  
 round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Середня квадратична помилка =",  
 round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Середня абсолютна помилка =",  
 round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Пояснена оцінка дисперсії =",  
 round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 оцінка =", round(sm.r2\_score(y\_test, y\_test\_pred), 3))  
  
output\_model\_file = 'model.pkl'  
  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nНова середня абсолютна помилка =",  
 round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))  
  
# Поліноміальна регресія  
polynomial = PolynomialFeatures(degree=10)  
X\_train\_transformed = polynomial.fit\_transform(X\_train)  
  
datapoint = [[7.75, 6.35, 5.56]]  
poly\_datapoint = polynomial.fit\_transform(datapoint)  
  
poly\_linear\_model = linear\_model.LinearRegression()  
poly\_linear\_model.fit(X\_train\_transformed, y\_train)  
print("\nLinear regression:\n",  
 regressor.predict(datapoint))  
print("\nPolynomial regression:\n",  
 poly\_linear\_model.predict(poly\_datapoint))

Text

Description automatically generated

Рисунок 5. Результат виконання

Завдання 4 Регресія багатьох змінних

*Лістинг додатку*

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model  
from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error  
  
# Завантажимо набір данних  
diabetes\_X, diabetes\_y = datasets.load\_diabetes(return\_X\_y=True)  
  
# Використаємо лише одну "розмірність"  
diabetes\_X = diabetes\_X[:, np.newaxis, 2]  
  
# Розіб'ємо дані на тестові та тренувальні  
diabetes\_X\_train = diabetes\_X[:-20]  
diabetes\_X\_test = diabetes\_X[-20:]  
  
diabetes\_y\_train = diabetes\_y[:-20]  
diabetes\_y\_test = diabetes\_y[-20:]  
  
# Створимо модель лінійної регресії  
regr = linear\_model.LinearRegression()  
  
# Навчимо  
regr.fit(diabetes\_X\_train, diabetes\_y\_train)  
  
# Зробимо передбачення  
diabetes\_y\_pred = regr.predict(diabetes\_X\_test)  
  
# Коефициенти  
print("Regression coef: \n", regr.coef\_)  
print("Regression intercept: \n", regr.intercept\_)  
# Середня абсолютна похибка  
print("Mean absolute error :",  
 round(mean\_absolute\_error(diabetes\_y\_test, diabetes\_y\_pred), 2))  
# Помилка середньої похибки  
print("Mean squared error: %.2f" % mean\_squared\_error(diabetes\_y\_test, diabetes\_y\_pred))  
print("R2 score: %.2f" % r2\_score(diabetes\_y\_test, diabetes\_y\_pred))  
  
  
fig, ax = plt.subplots()  
ax.scatter(diabetes\_y\_test, diabetes\_y\_pred, edgecolors=(0, 0, 0))  
ax.plot([diabetes\_y.min(), diabetes\_y.max()], [diabetes\_y.min(), diabetes\_y.max()], 'k--', lw=4)  
ax.set\_xlabel('Виміряно')  
ax.set\_ylabel('Передбачено')  
plt.show()

Chart, scatter chart

Description automatically generated

Рисунок 6. Результат виконання

Завдання 5. Самостійна побудова регресії

Варіант 1

m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)

*Лістинг програми*

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import PolynomialFeatures  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
  
X = X.reshape(-1, 1)  
Y = y.reshape(-1, 1)  
  
lin = LinearRegression()  
lin.fit(X, y)  
  
poly = PolynomialFeatures(degree=2)  
X\_poly = poly.fit\_transform(X)  
  
poly.fit(X\_poly, y)  
lin2 = LinearRegression()  
lin2.fit(X\_poly, y)  
  
Y\_NEW = lin2.predict(X\_poly)  
r2 = r2\_score(Y, Y\_NEW)  
  
print('R2: ', r2)  
  
# Visualising the Linear Regression results  
plt.scatter(X, y, color='blue')  
plt.plot(X, lin.predict(X), color='red')  
plt.title('Linear Regression')  
plt.show()  
  
# Visualising the Polynomial Regression results  
plt.scatter(X, y, color='blue')  
plt.plot(X, lin2.predict(poly.fit\_transform(X)), color='red')  
plt.title('Polynomial Regression')  
plt.show()

Chart, scatter chart

Description automatically generatedChart

Description automatically generated



Рисунок 7. Графіки Лінійної та Полімінальної регресії та R2 оцінка

Побудована модель є середньою через значне відхилення від осей.

Завдання 2.6. Побудова кривих навчання

*Лістинг програми:*

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.model\_selection import train\_test\_split  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import PolynomialFeatures  
  
  
def plot\_learning\_curves(model, X, y):  
 X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2)  
 train\_errors, val\_errors = [], []  
 for m in range(1, len(X\_train)):  
 model.fit(X\_train[:m], y\_train[:m])  
 y\_train\_predict = model.predict(X\_train[:m])  
 y\_val\_predict = model.predict(X\_val)  
 train\_errors.append(mean\_squared\_error(y\_train\_predict, y\_train[:m]))  
 val\_errors.append(mean\_squared\_error(y\_val\_predict, y\_val))  
 plt.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label='train')  
 plt.plot(np.sqrt(val\_errors), "b-", linewidth=3, label='val')  
 plt.show()  
  
  
m = 100  
X = np.linspace(-3, 3, m)  
y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, m)  
  
X = X.reshape(-1, 1)  
Y = y.reshape(-1, 1)  
  
lin = LinearRegression()  
lin.fit(X, y)  
  
poly = PolynomialFeatures(degree=2, include\_bias=False)  
X\_poly = poly.fit\_transform(X)  
  
poly.fit(X\_poly, y)  
lin2 = LinearRegression()  
lin2.fit(X\_poly, y)  
  
Y\_NEW = lin2.predict(X\_poly)  
r2 = r2\_score(Y, Y\_NEW)  
  
  
polynomial\_regg = Pipeline([  
 # ("poly\_features", PolynomialFeatures(degree=2, include\_bias=False)),  
 ("poly\_features", PolynomialFeatures(degree=10, include\_bias=False)),  
 ("lin\_reg", LinearRegression()),  
])  
  
plot\_learning\_curves(polynomial\_regg, X, y)

Chart, line chart

Description automatically generatedChart

Description automatically generated

Рисунок 8. Крива навчання для моделі за ступенем 2(зліва) та 10(справа)

**Висновок:** Розглянули особливості методів регресії та побудови графіків на основі отриманих даних.